GRAPHITE: A Visual Query System for Large Graphs

Duen Horng Chau, Christos Faloutsos,
Hanghang Tong, Jason I. Hong
Carnegie Mellon University
{dchau, christos, htong, jasonh}@cs.cmu.edu

Brian Gallagher, Tina Eliassi-Rad
Lawrence Livermore National Laboratory
{bgallagher, eliassirad1}@llnl.gov

Abstract

We present GRAPHITE*, a system that allows the user to visually construct a query pattern, finds both its exact and approximate matching subgraphs in large attributed graphs, and visualizes the matches. For example, in a social network where a person's occupation is an attribute, the user can draw a 'star' query for "finding a CEO who has interacted with a Secretary, a Manager, and an Accountant, or a structure very similar to this".* GRAPHITE *uses the* G-Ray *algorithm to run the query against a user-chosen data graph, gaining all of its benefits, namely its high speed, scalability, and its ability to find both exact and near matches. Therefore, for the example above,* GRAPHITE *tolerates indirect paths between, say, the CEO and the Accountant, when no direct path exists.* GRAPHITE *uses fast algorithms to estimate node proximities when finding matches, enabling it to scale well with the graph database size.*

We demonstrate GRAPHITE*'s usage and benefits using the DBLP author-publication graph, which consists of 356K nodes and 1.9M edges. A demo video of* GRAPHITE *can be downloaded at* `http://www.cs.cmu.edu/~dchau/graphite/graphite.mov`.

Figure 1. The GRAPHITE **user interface showing the query pattern (left) for a chain of authors from four different conferences. Nodes are authors; attributes are conferences; edges indicate co-authorship. One best-effort match (right) is Indyk (STOC), Muthu (SIGMOD), Garofalakis bridging Muthu and Jordan (ICML), and Hinton bridging Jordan and Fels (ISBMS).**

1. Introduction

People often want to find patterns in graphs, such as social networks, to better understand their dynamics. One such use is to spot anomalies. For example, in social networks where a person's occupation is an attribute, we might want to find money laundering rings that consist of alternating businessmen and bankers. But, then, we face several challenges: (1) we need a conveninent way to specify this ring pattern as a query, with appropriate attributes (e.g., businessman, banker) assigned to each node; (2) we need to find all potential matches for this pattern; we want near matches as well, such as allowing another person between a businessman and a banker, because we may not know the *exact* structure of a money laundering ring; (3) the graph matching process should be fast, avoiding expensive operations, such as joins; (4) we want to visualize all the matches to better interpret them.

We present GRAPHITE, a system designed to solve the above challenges. GRAPHITE stands for **Graph** **I**nvestigation by **T**opological **E**xample. It provides a usable integrated environment for handling the com-

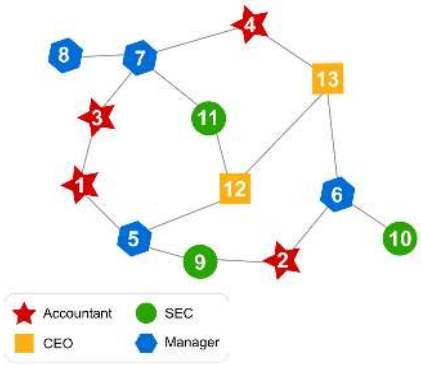

Figure 2. A ficticious network of people, whose job titles (attributes) are represented by shapes and colors.

(a) Loop query (b) A matching subgraph

Figure 3. A loop query and a match

plete workflow of querying a large graph for subgraph patterns. Users can (1) naturally draw the structure of the pattern they want to find and assign attribute values to nodes; (2) run the query against a user-chosen data graph, using the *G-Ray* method, to quickly locate exact and near matches; (3) obtain matches in a matter of seconds; and (4) visualize the matches.

Figure 1 is a screenshot of GRAPHITE when we ask for a chain of four coauthors in DBLP: a STOC'05 author, a SIGMOD'06 author, an ICML'93 author, and an ISBM'05 author. Such a chain does not exist, but GRAPHITE returns a best-effort match, with two intermediate nodes (in white): Minos Garofalakis, who bridges Muthu (SIGMOD) with Jordan (ICML, a premier machine learning conference) and Geoffrey Hinton, who bridges Michael Jordan (ICML) and Sidney Fels (ISBMS, a conference on biomedical simulation).

This paper is organized as follows. Section 2 gives the formal definition of our subgraph matching problem. Section 3 describes the system details. Section 4 describes what we will be demonstrating for GRAPHITE. Section 5 discusses related work. We conclude our contributions in Section 6.

2. Problem Definition

We describe the subgraph matching problem that GRAPHITE is designed to solve. Consider the fictitious social network in Figure 2, where nodes are people, whose attributes (job titles) are represented by shapes and colors. We define the problem as: *given*

- a data graph (e.g., Figure 2), where the nodes have one categorical attribute, such as job titles,
- a query subgraph describing the configuration of nodes that the user wants to find (e.g., Figure 3(a)), and
- the number of desired matching subgraphs k,

find k matching subgraphs, that match the query as well as possible.

For inexact matches, they should be ranked accordingly to their quality, such as how "similar" they look to the query. Incidentally, since we are using the *G-Ray* algorithm, the matching subgraphs will be automatically ranked according to its goodness function, giving convincing and intuitive rankings [6].

3. Introducing Graphite

GRAPHITE is a system for visually querying large social networks through direct manipulation, finding exact and near matches, and visualizing them.

The User Interface and Interactions. Figure 4 shows GRAPHITE's user interface. The left half is the *query area* (a), where users draw their query subgraphs. They can assign an attribute to a node by double-clicking on it and picking a value from a pop-up dialog (f). Users can create nodes and edges with the *editing control* (middle icon at (c)), reposition or delete them with the *picking control* (arrow icon at (c)), pan around the view port with the *panning control* (hand icon at (c)), and zoom in or out with the mouse scroll wheel. The right half of the user interface is the *results area* (b), which shows the exact and near matches as tabs (e) that the user can inspect conveniently by flipping through them. Users can specify the number of matches they want to find with the text box at the bottom of the interface (d). They can then click the *Find Matches* button to start the pattern matching process.

Algorithm for Finding Matches. There are many different subgraph matching algorithms that could be used for GRAPHITE; if we only wanted exact matches, we could write SQL queries to specify the query patterns. However, we chose the *G-Ray* algorithm for the following two advantages. First, when no exact matches exist, it automatically searches for best-effort matches (tolerating longer, indirect paths). Second, thanks to its proposed goodness function [6], it ranks the resulting matches, returning results that are

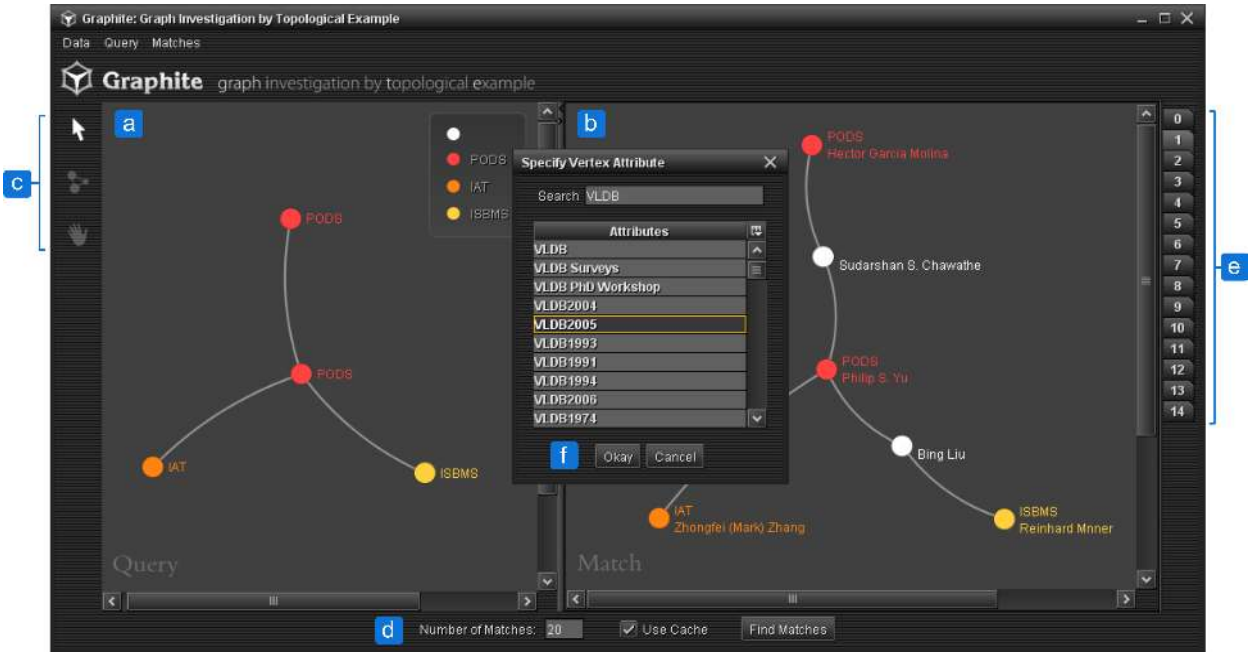

Figure 4. The GRAPHITE user interface. (a) User-specified 'star' query pattern. (b) Near match for the 'star' pattern. Nodes are authors; attributes are conferences; edges link co-authors. The query asks for an author who has published in PODS, with connections to authors of IAT, PODS, and ISBMS. (c) Users can select, create, move and delete nodes and edges; they can also zoom and pan. (d) Users specify number of matches. (e) Matches shown as tabs. (f) Users double-click a node to bring up a dialog for filtering attributes down to the ones that contain the filtering text.

empirically more important to the users, thus avoids flooding the user with a potentially huge number of less important matches.

Implementation. GRAPHITE is a Java SE 6 application. It uses the JUNG[1] Java library for editing and visualizing graphs. *G-Ray*, the backend algorithm that GRAPHITE uses for subgraph matching is written in the MATLAB programming language. GRAPHITE uses the RemoteMatLab software library[2] to remotely call into an instance of MATLAB that has been started as a server, passing query patterns to the algorithm and obtaining matches from it.

4. Demonstration

Datasets. We use the DBLP dataset,[3] from which we construct an attributed graph where each node is an author and the node's attribute is the combination of a conference name and a year (e.g., "ICDM 2008"). We describe this attributed graph by two matrices: (1) a node-to-node matrix, which represents the co-authorship among authors where entry (i, j) is the number of coauthored papers between author i and j; and (2) a node-to-attribute matrix, which represents the author-conference relationship where entry (i, j) equals 1 if author i has published in conference j, and 0 otherwise. In total, there are 356,364 nodes, 1,905,970 edges, and 12,920 possible attribute values.

Demonstration Details. We will demonstrate how to draw common query structures, such as a 'line' pattern (as in Figure 1, discussed in Section 1), and a 'star' pattern (as in Figure 4). Our audience can also create their own query patterns. The 'star' query asks for an author who has published in PODS (in red), who has co-authored papers with three other authors from the conferences IAT (orange), PODS (red), and ISBMS (yellow). In one of the highest-ranking matches (on the right), the PODS author in the center is Philip Yu, a prolific author in databases and data mining. The other PODS author is Hector Garcia-Molina, also extremely prolific, with an indirect con-

[1] `http://jung.sourceforge.net/`

[2] `http://plasmapowered.com/wiki/index.php/Calling_MatLab_from_Java`

[3] `http://www.informatik.uni-trier.de/~ley/db/`

nection to Philip through Chawathe, his ex-advisee. Zhongfei (Mark) Zhang is the matching author for IAT, Intelligent Agent Technology, who is a computer vision researcher with a recent interest in data mining, hence the connection to Philip Yu.

We will show our audience how to assign attributes to the query nodes, via the dialog shown in Figure 4(f), which quickly filters possible attribute values down to the ones that contain the filtering text. We will also perform real-time pattern matching for the query patterns by communicating with the backend Matlab server. We will engage our audience to make sense of the exact and near matches that GRAPHITE displays, and to offer their feedback on the quality of the results.

5. Related Work

Graph matching algorithms vary widely due to differences in the specific problems they address. *G-Ray* is a fast approximate algorithm for inexact pattern matching in large, attributed graphs. It extends the ideas of connection subgraphs [2] and centerpiece graphs [5] and applies them to pattern matching in attributed graphs. This work is also related to the idea of network proximity, which builds on connection subgraphs [3].

Our work focuses on finding instances of user-specified patterns in graphs. Graph mining work in the database literature focuses on related problems, like the discovery of frequent or interesting patterns [7], near-cliques [4], and inexact querying of databases [1]. However, none of these methods can do 'best-effort' matching for arbitrary shapes, like loops, that GRAPHITE can handle.

6. Conclusions

We have presented GRAPHITE, a system for visually querying large graphs. GRAPHITE's contributions include (1) providing an integrated environment for handling the complete workflow of querying a large graph for subgraph patterns; (2) providing an intuitive means for users to specify query patterns by simply drawing them; (3) finding and ranking both exact and near matches, using the best-effort *G-Ray* algorithm; (4) visualizing matches to assist users in understanding and interpreting the results; and (5) delivering results in high speed for large graphs (such as the DBLP graph, consisting of 356K nodes), returning results in seconds, on a commodity PC.

We believe GRAPHITE can become a useful tool for scientists and analysts working on graph problems to quickly find patterns of their choosing, to experiment with and to confirm their speculations, and to better understand the dynamics of their graphs.

7 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. IIS-0705359 and under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 . Duen Horng Chau is supported by Symantec Research Labs Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

[1] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, pages 431–440, 2002.

[2] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 118127, 2004.

[3] Y. Koren, S. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–255, 2006.

[4] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD '05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[5] H. Tong and C. Faloutsos. Center-piece subgraphs: Problem definition and fast solutions. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–413, 2006.

[6] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–746, New York, NY, USA, 2007. ACM.

[7] X. Yan, P. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *ICDM '04: Proceedings of the 4th International Conference on Data Mining*, pages 335–346, 2004.